RMIT at the TREC 2015 LiveQA Track

(Authors listed in lexicographical order of the surnames)
Ruey-Cheng Chen, J. Shane Culpepper, Tadele Tadela Damessie, Timothy Jones,
Ahmed Mourad, Kevin Ong, Falk Scholer, Evi Yulianti
RMIT University
Melbourne, Australia
{*ruey-cheng.chen, shane.culpepper*}*@rmit.edu.au, s3497203@student.rmit.edu.au,*
{*timothy.jones, ahmed.mourad, kevin.ong, falk.scholer, evi.yulianti*}*@rmit.edu.au*

***Abstract*—This paper describes the four systems RMIT fielded for the TREC 2015 LiveQA task and the associated experiments. The challenge results show that the base run RMIT-0 has achieved an above-average performance, but other attempted improvements have all resulted in decreased retrieval effectiveness.**

***Keywords*-TREC LiveQA 2015; RMIT; passage retrieval; summarization; query trimming; headword expansion**

I. Overview

In the TREC LiveQA 2015 challenge, we experimented with four different retrieval-based answer-finding strategies. Instead of pursuing a traditional question-answering approach that seeks deeper understanding of the questions, we focused on simple enhancements, such as summarization, query trimming, and headword expansion. These are common techniques used in IR, and can easily be integrated into research-purpose retrieval engines or pipelines of a similar scale. In our experiments, we considered the following research questions:

RQ 1:. *Will shorter or longer summaries result in better answers?*

RQ 2:. *Should all of the terms in a question be used, or should only a subset of "important" terms be used?*

RQ 3:. *Can headword expansion using external resources improve the quality of answers?*

To answer these questions, we configured four different systems in the 2015 challenge. Surprisingly, we found that passage retrieval using the full query with minimal summarization and no query reduction or expansion produced the best results.

II. Data and Retrieval Settings

We now describe the collection and retrieval setting used in our system.

A. Server architecture

The servers are built on top of the computing resources we allocated from NecTAR,[1] the Australian National Research cloud computing network. Throughout the challenge, we use only one instance to host all of the services.

[1]https://www.nectar.org.au

Question responses were dependent on the server ID when the questions were served (see Figure II). In the most basic form, the server converted the questions into a bag-of-words query, and ran these against the indexes. The three most relevant passages retrieved are then submitted to our summarizer component, which outputs a predefined number of sentences ranked by relevance within the summarizer (see Section II-F).

In our final iteration, we wanted to see if a subset of "important" terms derived from a headword expansion would improve performance. In this iteration, we extracted key terms from the original question, which were then trimmed. Query expansion of the headwords was carried out using word2vec. The reduced query with word2vec headword expansion is passed to the query processor. The retrieved documents were then summarized by the summarizer, and the first sentence is then returned as the response.

The various services were connected using a resource allocator written in the Go Programming Language. It included graceful handling of timeouts, and guaranteed responses within the 60 second window. For the curious reader, our code is available under a BSD license.[2] Please cite this paper if you use the code for anything.

B. Run descriptions

RMIT-0[3] **(automatic):** Indri bag-of-words passage retrieval using all of the terms in the question title, and top three passages summarized by the method described in Section II-F.

RMIT-1 (automatic): Indri bag-of-words passage retrieval using all of the terms in the question title, and the top three passages summarized by the method described in Section II-F. However, only the first sentence generated by the summary process was returned.

RMIT-2 (automatic): Indri bag-of-words passage retrieval using terms derived from the question title, term trimming (down to 5 terms), headword expansion (adding up to 5 terms) using word2vec as described in Section II-H, and the top three passages summarized by the method described in Section II-F. Only the first sentence generated by the summary process was returned.

[2]https://github.com/TimothyJones/trec-liveqa-server

[3]Originally referred to as Monash-System2 in the LiveQA challenge.

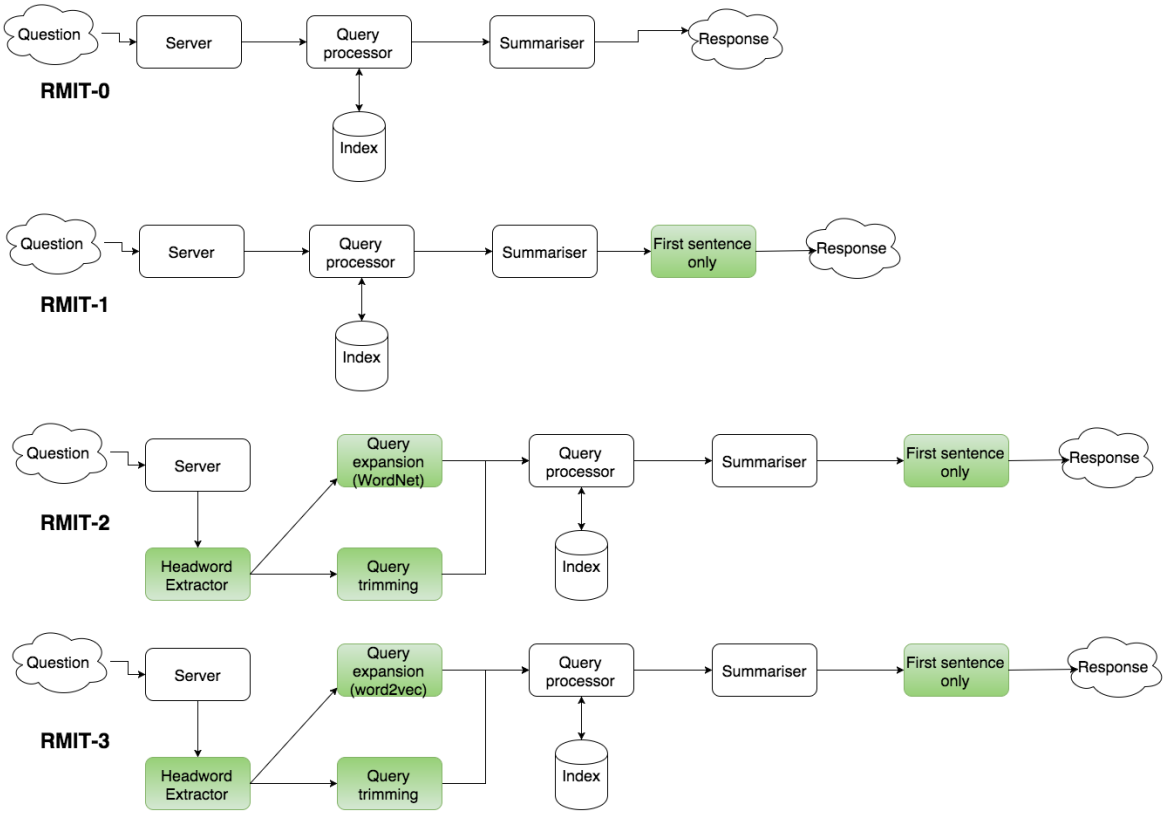

Figure 1. System architecture for each RMIT system. Green shading indicates components that are different when compared to RMIT-0.

RMIT-3 (automatic): Indri bag-of-words passage retrieval using terms derived from the question title, term trimming (down to 5 terms), headword expansion (adding up to 5 terms) using wordnet as described in Section II-H, and top three passages summarized by the method described in Section II-F. Only the first sentence generated by the summary process was returned.

C. Collection

Table I summarizes the collections we used in the task. We indexed AQUAINT and AQUAINT-2 as TREC Text documents. To prepare the data set for English Wikipedia, we used the open-source tool wp-download, [4] to fetch a dump,[5] extracted all of the XML content using wikiextractor,[6] and indexed every Wikipedia page as a document. To index the Yahoo! Answers CQA data, we processed the collection as follows: rather than just indexing the best answers, we extracted and indexed only the answers to previous questions and stored them as documents. We did not make use of the subject and content tags (i.e., question title and description) in the data.

D. Indexing

We used Indri 5.9 as our retrieval engine with Krovetz stemming and the default InQuery stoplist.[7] Once the collection described in the previous section was indexed, we ended up with a single 39 GB index that contained 38.7 million documents and 12.2 million unique terms.

E. Passage Retrieval

According to our prior tests over the live stream, the question title contains 10 terms on average, and the body size is around 30 terms. However, as some of the question bodies can be quite long, we decided to construct queries using only the title. In two of our submitted runs, we tried different ways of expressing the same query intent by doing query reduction and expansion. The details of this approach are described in Section II-B.

We used the fixed-sized passage operator `#combine[passage100:50](...)` provided by Indri to retrieve and parse the top three passages from document texts. The result was then sent to the summarizer. For performance reasons, and the length of some of the queries, we used a bag-of-words query, and BM25 ranking. For BM25, our parameter configuration was $k_1 = 0.9$ and $b = 0.4$.[8]

F. Summarization

For summarization, we used the model proposed by Takamura and Okumura [7] to generate extractive summaries from the top-ranked passages. In this model,

[4]https://github.com/babilen/wp-download
[5]We used an enwiki dump produced on May 15, 2015.
[6]https://github.com/attardi/wikiextractor
[7]http://www.lemurproject.org/indri.php

[8]The values for b and k_1 are different than the defaults reported by Robertson et al. [5]. These parameter choices were reported for Atire and Lucene in the 2015 IR-Reproducibility Challenge, see github.com/lintool/IR-Reproducibility for further details.

Table I
SUMMARY OF COLLECTIONS INDEXED TO ANSWER QUESTIONS.

Collection	Number of Documents	Number of Words	Description
AQUAINT	1,034K	506M	Newswire, 1999 - 2000
AQUAINT2	907K	410M	Newswire, Oct 2004 - Mar 2006
Wikipedia-EN	4,847K	1,775M	Online knowledge base
Yahoo! Answers CQA v1.0	31,972K	1,462M	Question answers converted to documents from the Yahoo! Answers website.

summarization is characterized as a two-way optimization problem, in which coverage over important words is maximized, and redundancies are minimized simultaneously. The mathematical formulation is given as follows:

$$\begin{aligned} \text{maximize} \quad & (1-\lambda)\sum_j w_j z_j + \lambda \sum_i \sum_j x_i w_j a_{ij} \\ \text{subject to} \quad & x_i \in \{0,1\} \text{for all } i; \\ & z_j \in \{0,1\} \text{for all } j; \\ & \sum_i c_i x_i \leq K; \\ & \sum_i a_{ij} x_i \geq z_j \text{for all } j \end{aligned} \tag{1}$$

To produce an extractive summary, one basically makes a choice over the set of sentences and decides what to include. By doing so, one also makes an implicit choice over words. This choice is modeled in the optimization problem as two sets of variables x_i and z_j, the former indicating the binary decision on keeping sentence i, and the latter on keeping word j in the summary. In other words, for each sentence i, x_i is set to 1 if sentence i is to be included in the summary, or 0 otherwise. Analogously for each term j, z_j is set to 1 if term j is included.

In this problem, c_i denotes the cost of selecting sentence s_i (i.e. number of characters in s_i), and w_j denotes the weight of word j. We used a TF-IDF weighting scheme in which the term frequency (*tf*) is derived from the question title and body, and the inverse document-frequency (*idf*) is learned from a background corpus. The term frequency collected from the question body is further penalized with a factor $\alpha < 1$ as the information given in the question body can be less precise than in the title.

$$w_j = \left[tf_{title}(j) + \alpha \; tf_{body}(j) \right] * idf(j) \tag{2}$$

The correspondence between the sentence i and the word j is coded in the indicator variable a_{ij}, whose value is set to 1 if the word j appears in sentence i, and 0 otherwise. With the first constraint, we limit the size of the summary to K characters at most (K is set to 1,000 throughout). With the second constraint, the word coverage is related to the sentence coverage, thus completing the formulation.

Empirically, we fine-tuned the parameters λ and α based on prior test runs. In the challenge, we set $\lambda = 0.1$ and $\alpha = 0.43$. We used the IBM CPLEX solver to compute the optimal allocation.

G. Headword Detection

A *headword* is the key term in the question that helps to retrieve the most relevant documents. For example, consider the question: `What are the sales goals daily and monthly at MAC Cosmetics?`. Here, the headword is `sales`. The process of generating the hypernyms is divided into two stages. To extract the headword, we generate the syntactic parse tree of the question using the Stanford parser. Then, we apply the rule-based model initially defined by Collins [1] and later refined by Huang et al. [2] and Silva et al. [6].

H. Query Trimming and Headword Expansion

We also experimented with a feature called query trimming, which is to use only the "important" terms in the question title to retrieve answers. The questions from Yahoo! Answers are quite verbose, and intended to be human readable. However, verbose queries are known to perform poorly in many keyword-based IR systems. So, it seems sensible to use only a subset of terms from the question, especially if the terms selected are likely to contribute the most in the ranking function.

First, we used the WAND implementation from Petri et al. [3, 4] to extract the MaxScore U_b for each term.[9] The MaxScore list is then loaded into memory when the server starts. At query time, we used the list to order terms by impact, and trim the initial query down to a predefined size. The size is set to five terms throughout the experiments where trimming is applied.

We also experimented with two ways of expanding the headword term in each query externally, by drawing information from resources such as word2vec and wordnet:

- word2vec: In this method, we took a pre-trained word embedding model distributed with word2vec[10] and used gensim[11] to populate a list of query terms that are most similar to a given input.
- wordnet: In this method, we implemented the models proposed by Huang et al. [2] and Silva et al. [6]. We extract the hypernyms of the head word using WordNet, and map the Penn Treebank POS Tags to WordNet tags to decide which part of speech senses should be considered. Then, following the algorithm of Huang et al. [2] for head word-sense disambiguation, we calculate terms overlap between the definition of each sense and the definition of

[9]The code is available at https://www.github.com/jsc/WANDbl.
[10]https://code.google.com/p/word2vec
[11]https://radimrehurek.com/gensim

Table II
EFFECTIVENESS SUMMARY FOR ALL FOUR RMIT SYSTEMS WHEN COMPARED TO THE AVERAGE ACROSS ALL SYSTEMS PARTICIPATING IN THE 2015 LIVEQA TRACK.

Run ID	Avg. Score (0-3)	Success				Precision		
		@1+	@2+	@3+	@4+	@2+	@3+	@4+
RMIT0	**0.663**	0.987	**0.364**	**0.220**	**0.082**	**0.369**	**0.223**	**0.083**
RMIT1	0.435	0.992	0.267	0.130	0.039	0.269	0.131	0.039
RMIT2	0.378	**0.998**	0.232	0.115	0.034	0.232	0.115	0.034
RMIT3	0.412	0.994	0.251	0.126	0.038	0.252	0.127	0.038
All Runs	0.465	0.925	0.262	0.146	0.060	0.284	0.159	0.065

context words (each word in the question excluding the head word) with maximum depth of six. The optimal sense (the one which results in the maximum number of common words) is chosen to populate the list of synonyms.

Note that in the latter method, we use the context of the question, excluding the headword, to resolve the semantic ambiguity of different senses and populate a list of synonyms. For example, consider again the question *What are the sales goals daily and monthly at MAC Cosmetics?*, the headword is `sales` and the hypernyms are `gross sales`, `income`, `financial gain`, and `gain sum`.

For either approach, we added the top five generated expansion terms back into the query without any modification. Note that when query trimming is applied, the query would first get trimmed down to 5 terms, and then expanded using the headword to at most 10 terms totally.

III. RESULTS

The LiveQA challenge results are given in Table II, where our submitted runs and the average result across all runs are shown. Our base run RMIT-0 delivered the best performance in our experiment, achieving 0.663 in Avg Score. The base run outperforms the average across all runs submitted to the challenge. On the other hand, all refinements that we tested resulted in decreased performance, below the average over all submitted runs.

Limiting the output to only the first sentence in the summary (RMIT-1) appears to have a negative effect on precision at all relevance level. This is surprising, since adding more sentences increases the likelihood that non-relevant information would appear in the summary.

Our result on query trimming and headword expansion also shows no improvement. Therefore, this combined strategy is not effective when top-sentence precision is of concern. Among the two expansion methods, word2vec (RMIT-2) appears to do more harm than wordnet (RMIT-3). We speculated that headword expansion using word2vec is not practically useful, as word2vec can be too aggressive sometimes, generating terms that are not synonyms to the headword and thus wildly biasing the original intent.

IV. CONCLUSION

We have explored four different system configurations for the TREC LIVEQA Track in 2015. While we are pleasantly surprised with the performance of our baseline system, we believe further improvements can still be realized using query reduction and headword expansion. We hope that further post-run analysis will provide insight into why the approaches were not successful in our current system configurations.

Acknowledgment. This work was supported by the Australian Research Council's *Discovery Projects* Scheme (DP140102655). Shane Culpepper is the recipient of an Australian Research Council DECRA Research Fellowship (DE140100275).

REFERENCES

[1] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[2] Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics, 2008.

[3] Matthias Petri, J Shane Culpepper, and Alistair Moffat. Exploring the magic of wand. In *Proceedings of the 18th Australasian Document Computing Symposium*, pages 58–65. ACM, 2013.

[4] Matthias Petri, Alistair Moffat, and J Shane Culpepper. Score-safe term-dependency processing with hybrid indexes. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 899–902. ACM, 2014.

[5] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. TREC-3*, 1994.

[6] Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

[7] Hiroya Takamura and Manabu Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proc. of EACL*, pages 781–789, 2009.